UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

APLICACIÓN PARA RACIONALIZAR EL USO DEL MÓVIL

Rodrigo Alonso García-Amorena
Tutor: Carlos Aguirre Maeso

MAYO 2019

APLICACIÓN PARA RACIONALIZAR EL USO DEL MÓVIL

AUTOR: Rodrigo Alonso García-Amorena
TUTOR: Carlos Aguirre Maeso

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2019

Resumen (Castellano)

Este Trabajo de Fin de Grado agrupa una serie de tecnologías distintas para crear una aplicación Android que tiene como objetivo incentivar en el usuario la disminución de su uso del teléfono móvil. Esto es porque hoy en día estos dispositivos se utilizan, en mi opinión, de manera desmedida e innecesaria en la mayoría de los casos, y muchos usuarios tan solo necesitan y/o desean un pequeño aliciente para dejar de hacerlo.

Si bien es cierto que el lector se podría plantear si la mejor alternativa para evitar el uso del móvil es una aplicación, lo cual es un tanto paradójico, mi respuesta ante eso es que es un mal necesario para lograr un bien superior.

La aplicación pretende lograr estos objetivos a través de un acuario realizado en el motor de videojuegos Unity. El contenido de este acuario progresa en conformidad con el uso del dispositivo a través de una biblioteca de Android desarrollada en Java que accede al tiempo de uso de todas las demás aplicaciones. Esta información se procesa con un algoritmo cuya salida decide la medida en la que el acuario varía, tanto para bien, aumentando el tamaño de los peces, como para mal, disminuyéndolo.

Los datos de un usuario serán persistentes ya que éste se autenticará en Google Play Juegos y se mantendrá constancia de sus pertenencias en una base de datos online y en tiempo real en Firebase, de Google.

Todo lo anterior crea un entorno que permite poder compartir tu acuario con otras personas y crear competencia entre acuarios, en otras palabras, competencia en reducir el tiempo de uso de los aparatos Android.

Palabras clave (castellano)

Acuario, tiempo, Android, Firebase, Google Play Juegos, Unity, aplicación, móvil

Abstract (English)

This Bachelor Thesis groups a series of different technologies to create an Android application that aims to encourage the user to reduce their use of the mobile phone. This is because today these devices are used, in my opinion, in an excessive and unnecessary way in most cases, and many users just need and/or want a small incentive to stop doing so.

While it is true that the reader could consider whether the best alternative to avoid the use of mobile is an application, which is somewhat paradoxical, my response to that is that it is a necessary bad to achieve a higher benefit.

The application aims to achieve these goals through an aquarium made in the Unity video game engine. The content of this aquarium progresses in accordance with the use of the device through an Android library developed in Java that accesses the time of use of all other applications. This information is processed with an algorithm whose output decides the extent to which the aquarium varies, both for good, increasing the size of the fish, or for worse, decreasing it.

The data of a user will be persistent as he will be authenticated in Google Play Games and his belongings will be kept in a record on an online database and in real time in Firebase, Google.

All the above creates an environment that allows you to share your aquarium with other people and create competition between aquariums, that is, in other words, competence in reducing the time of use of the Android devices.

Keywords (inglés)

Aquarium, time, Android, Firebase, Google Play Games, Unity, application, smartphone

Agradecimientos

Quiero agradecer a todo aquel que ha escuchado la idea del proyecto con especial mención a los que se han mojado aportando ideas y su opinión al respecto.

También quiero agradecer a los que han estado siempre disponibles para mí en este último año y durante el resto de la carrera, en los momentos buenos, pero sobre todo en los momentos peores.

Finalmente, agradezco a mi tutor haberme permitido realizar el proyecto. Por aceptar mi idea desbocada y permitirme libertad a la hora de elaborarla.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

En los días que corren, la tecnología ha evolucionado radicalmente, llegando a proveer muchas utilidades de manera barata e instantánea. Los teléfonos móviles son el claro ejemplo. Se prevé que en 2020 habrá 50 trillones de ellos. Estos dispositivos, aunque mantengan el nombre asociado a la utilidad de llamar, han aumentado considerablemente su abanico de posibilidades, tanto incluso que ésta ha pasado a segundo plano. A continuación, resumiré cuales son las utilidades más usadas que los móviles facilitan hoy en día.

Ilustración 1: Estadística del uso del móvil en un día

Hoy en día, la media de uso de los móviles excede las dos horas diarias. De ese tiempo la mayoría es utilizado en consultar páginas web. Curiosamente, las llamadas han bajado escalones en el uso hasta el quinto puesto.

En la mayoría de los casos, los móviles siguen manteniendo como funcionalidad prioritaria la comunicación, aunque hayan variado el medio para hacerlo: redes sociales, medios de difusión, chats, etc.

También por encima de las llamadas tenemos aplicaciones para jugar. En este sector cada vez son más los géneros y estilos de juego disponibles. Hace unos años estos juegos eran

inimaginables ni siquiera en dispositivos de mayor tamaño. Esto es debido a que, como es lógico, los terminales móviles cada vez albergan más potencia computacional.

Todas estas herramientas son armas de doble filo. Un mal uso podría convertir sus utilidades en dependencias. Tan diversas son las opciones que ofrecen que se podría usar el móvil casi en cada momento para descubrir, medir, entretener, comunicar, disfrutar y más.

Es por esto que pensé en desarrollar otra de estas herramientas, es decir, una aplicación, pero, que paradójicamente sirviese para racionalizar el uso de todas las demás. De manera que se utilicen de manera adecuada, racional y sin excesos. Recordemos que la virtud está en el punto medio.

1.2 Objetivos

Mi pretensión era hacer algo al respecto de esta utilización desmesurada e injustificada en el grueso de los casos de los dispositivos móviles. Pronto descubrí que existían otras aplicaciones que buscaban el mismo objetivo. Como suele pasar, la rueda ya estaba inventada en ese terreno, así que debía encontrar una manera diferente de formalizar mi idea sin reinventar aquella rueda.

Pronto tuve conocimiento de la existencia de una aplicación que bloqueaba el acceso al uso del teléfono durante un periodo que el usuario indicase. También sabía que había aplicaciones que medían el tiempo de uso diario del móvil y lo mostraban, meramente informativas.

Fue entonces cuando se me ocurrió diseñar una aplicación que premiara estas estadísticas, con un progreso visual y enseñable. Visual porque a través de la vista las cosas cobran un sentido único e irremplazable. Y enseñable para que surgieran piques entre los usuarios y éstos promovieran racionalizar aún más el uso del móvil. Es decir, dotar de un objetivo doble a los usuarios de la aplicación: aprovechar más su tiempo y mejorar su acuario como muestra de ese aprovechamiento, cosa que suele funcionar en el mercado actual.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción:**

Deja ver una primera visión del proyecto, la intención de su creación y los motivos de su autor.

- **Estado del arte:**

En él se citan y detallan las tecnologías empleadas para llevar a cabo el proyecto y la razón para usar cada una de ellas.

- **Diseño:**

Aquí se explica el objetivo final al que se pretende llegar y de qué manera se ha dividido y asignado cada subfunción a un módulo del proyecto.

- **Desarrollo:**

De forma más técnica, nos adentramos en los entresijos de cada módulo para ver qué funciones realizan y sobre todo cómo las realizan.

- **Integración, pruebas, resultados:**

Incluye cómo se relacionan los módulos entre sí y su cohesión para alcanzar el funcionamiento total de la aplicación. También se habla de las pruebas realizadas y el análisis de sus resultados.

- **Conclusiones y trabajo futuro:**

El título se auto explica. Aquí hablaré de las ideas que me ha aportado este proyecto tras su realización y de todas aquellas mejoras que deseo realizar después de presentar el trabajo.

- **Anexo:**

Se explicará brevemente cómo instalar la aplicación y cómo utilizarla.

2 Estado del arte

Para desarrollar esta aplicación y así poder alcanzar mis objetivos he usado diversas tecnologías. A continuación, explicaré cuales son y porqué las elegí.

2.1 Android

Ilustración 2: Logotipo de Android

Ya que se quiere restringir el uso de los móviles, en específico de los de sistema operativo Android, ha resultado necesario el aprendizaje de las herramientas principales que utilizan las aplicaciones desarrolladas para dispositivos de este tipo.

Uno de los retos de este TFG ha sido por tanto descubrir Android, que para mí era un entorno totalmente nuevo. Necesitaba entender con claridad sus entresijos ya que la aplicación que tenía en mente requería de su conocimiento íntegro: necesitaba acceder al contexto de la aplicación, solicitar permisos al usuario para que permitiera el acceso al tiempo de uso de otras aplicaciones, almacenar información de forma persistente en el móvil para que se mantuviera al cerrar la aplicación, aprender a fabricar mi propia biblioteca, etc.

Repasemos algunos de los conceptos propios de las aplicaciones Android con los que he tenido que familiarizarme y he utilizado para realizar el TFG:

- Los contextos y actividades: Un contexto, como sugiere el nombre, hace alusión al estado actual de la aplicación. Permite comprender a funciones u objetos ajenos lo que ha estado sucediendo previamente. Se suele usar para conocer el estado de una parte del programa, llamada actividad.

Bibliotecas de Android: Una biblioteca de Android está compuesta de los mismos elementos que una aplicación de Android. Puede disponer de todo lo necesario para compilar una aplicación: código fuente donde se encuentra la lógica escrita en Java o Kotlin, archivos de recursos escritos en XML que definen la interfaz y los elementos de diseño, y un archivo de manifiesto de Android, que otorga al sistema Android información imprescindible sobre tu aplicación para que la pueda ejecutar. Sin embargo, en vez de compilarse en un archivo APK que podría ejecutarse en un dispositivo, las bibliotecas de Android se compilan en archivos Android Archive (AAR) que pueden ser usados como dependencia en una aplicación de Android.

- Niveles de API: Desde el surgimiento del sistema operativo Android por parte de Google en 2008 ha visto múltiples actualizaciones y agregaciones de funcionalidad. Estas han ido recibiendo nombres de dulces en orden alfabético: Apple Pie, Banana Bread, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo y Pie.

Ilustración 3: Versiones de Android

Debido a la adición de nuevas funcionalidades en cada versión, un usuario con una versión anterior podría no poder ejecutar las funciones de una aplicación enfocada para una versión superior. Es por esto por lo que a la hora de escoger la versión (o nivel) de la API existe un compromiso entre número de usuarios que podrán usar la aplicación y las funcionalidades disponibles en cada nivel.

Un ejemplo de funcionalidad que requería un nivel mínimo de API de Android al que me he tenido que enfrentar es al obtener el servicio del sistema operativo encargado de las estadísticas de uso, a través de la función getSystemService, incorporada en la API número 23. Esto condicionó que descartara las versiones anteriores de API.

Nota: También existen bibliotecas de compatibilidad creadas por Google que ofrecen versiones de funciones nuevas que son compatibles con versiones anteriores. Estas son las llamadas *Compat*.

- Permisos: Una adversidad han sido los permisos de los que disponen las aplicaciones y que aumentan su radio de acción, existen tres clases:
    1. Los que son directamente accesibles como es el caso del acceso a internet, por ejemplo, y que solo hay que incluirlos en el archivo de manifiesto de Android y se otorgan automáticamente a la aplicación.
    2. Los permisos peligrosos, que comprometen la privacidad del usuario, como el uso de la cámara, los cuales requieren la dotación de ese permiso por parte del usuario. La aplicación en este tipo de permisos puede abrir una ventana que solicite dicha aprobación en tiempo de ejecución.
    3. Los permisos especiales, que no se pueden pedir en tiempo de ejecución de la aplicación, y todo lo que puedes hacer por el usuario es abrir la

sección de los ajustes pertinente para que pueda habilitarlos manualmente.

Lo has adivinado, los permisos para acceder al tiempo de uso de otras aplicaciones son del tercer tipo.

Hubo que modificar el archivo de manifiesto AndroidManifest.xml con el que Unity generaba la APK para que la aplicación saliera entre la lista de aplicaciones disponibles a las que se puede dar permisos de uso, llamados "android.permission.PACKAGE_USAGE_STATS".

- Preferencias de aplicación: Las preferencias de una aplicación, o preferencias compartidas entre las actividades de una aplicación son datos que una aplicación guarda para personalizar la experiencia del usuario. Estas se podrían almacenar perfectamente en una base de datos SQLite de las que provee Android para las aplicaciones. Sin embargo, Android proporciona otro método alternativo diseñado específicamente para administrar este tipo de datos de forma más sencilla y eficiente. Consiste en almacenar las preferencias en forma de clave-valor, es decir, cada una compuesta por un identificador único y un valor asociado a dicho identificador. Las preferencias se almacenan en ficheros XML.

El lenguaje de programación escogido para tratar con Android (en específico con la biblioteca de desarrollada) ha sido Java frente a Kotlin, que corre también sobre la máquina virtual de Java, debido a mi mayor conocimiento sobre el primero.

Ilustración 4: Logotipo de Java

A pesar de lo evidente que parece que se utilice una biblioteca de Android para una aplicación del mismo sistema, no lo es tanto. De hecho, el único motivo por el que se hace estrictamente necesario su uso es por el propósito de esta aplicación concreta.

Me adelanto brevemente al siguiente punto hablando de Unity, el motor de videojuegos que se ocupa de la parte gráfica del proyecto, para aclarar que éste normalmente posee las herramientas para ser autosuficiente en la plataforma de Android. Pero en este caso requerimos de un uso concreto. Esto es porque Unity carece de funciones específicas del sistema operativo de Android, por lo que necesita accederlas a través de una entidad externa que sí disponga de estas funciones. La biblioteca de Android escrita en Java, efectivamente, puede acceder al administrador de estadísticas de uso.

2.2 Unity

Ilustración 5: Logotipo de Unity

Unity es un motor de videojuegos para múltiples plataformas, en nuestro caso empleado para Android. El motor gráfico utiliza la librería de gráficos OpenGL entre otras. También utiliza el motor de físicas PhysX. Por último, el lenguaje de scripting principal de Unity en el que se codifica la lógica de los componentes es C#, que es un lenguaje de propósito general, de fuerte tipado, funcional y orientado a objetos y componentes.

Debido a que, de cara al usuario, la aplicación iba a ser puramente visual y que todos los beneficios se traducirían en mejoras de esa visualización, necesitaba un entorno que ofreciera comodidad y resolución en ese aspecto. Es por esto por lo que elegí Unity. Además, es ya un entorno familiar para mí, ya que he desarrollado previamente otras aplicaciones sobre él, aunque muy diferentes de la actual.

2.3 Google Play Juegos

Ilustración 6: Logotipo de Google Play Juegos

Google Play Juegos es una plataforma de videojuegos que incluye varios sistemas automatizados para que los desarrolladores puedan añadir un sistema de inicio de sesión a sus juegos, además de otras ventajas como logros, clasificaciones, estadísticas, etc.

El motivo principal por el que deseaba usar este tipo de autenticación es por aprovechar el nombre único que tiene cada usuario en esta plataforma para usarlo como identificador de usuario, y así poner su acuario en Firebase bajo ese identificador.

El segundo motivo era por darle categoría de juego, es decir, clasificarlo como actividad que ofrece diversión, pero a la vez retos y competencia.

Para usar este método de autenticación, incluso en modo depuración sin estar publicada la aplicación, se requería ser desarrollador en Google Play, lo cual exigía dar de alta una cuenta como desarrollador (por un coste de veinticinco euros) y añadir la aplicación en la

consola de desarrolladores de Google Play. La aplicación debía firmarse con una clave pública y una privada, y después con el programa keytool se debía convertir en huella digital SHA1 para poder certificar la aplicación.

2.4 Firebase

Ilustración 7: Logotipo de Firebase

De Firebase he usado dos de sus múltiples funciones:

- Firebase Realtime Database:

Es una base de datos en tiempo real, es decir, que se actualiza después de unos milisegundos de ser modificada por código y que, a través de escuchadores, puede actualizarse simultáneamente en múltiples dispositivos cuando se produce un cambio en ella. Esta base de datos utiliza el formato JSON, recurriendo al paradigma de un diccionario, con clave y valor.

La utilidad que le he dado en la aplicación es la de recoger los usuarios y sus acuarios almacenando por cada pez que posean, el progreso actual de ese pez y su tipo. De manera que sea accesible desde cualquier dispositivo con acceso a internet y en cualquier momento.

- Firebase Authentication:

Es la rama de Firebase que bloquea el acceso a la base de datos a usuarios no autenticados y que permite a usuarios autenticados obtener y modificar datos. Existen múltiples formas de autenticarse:

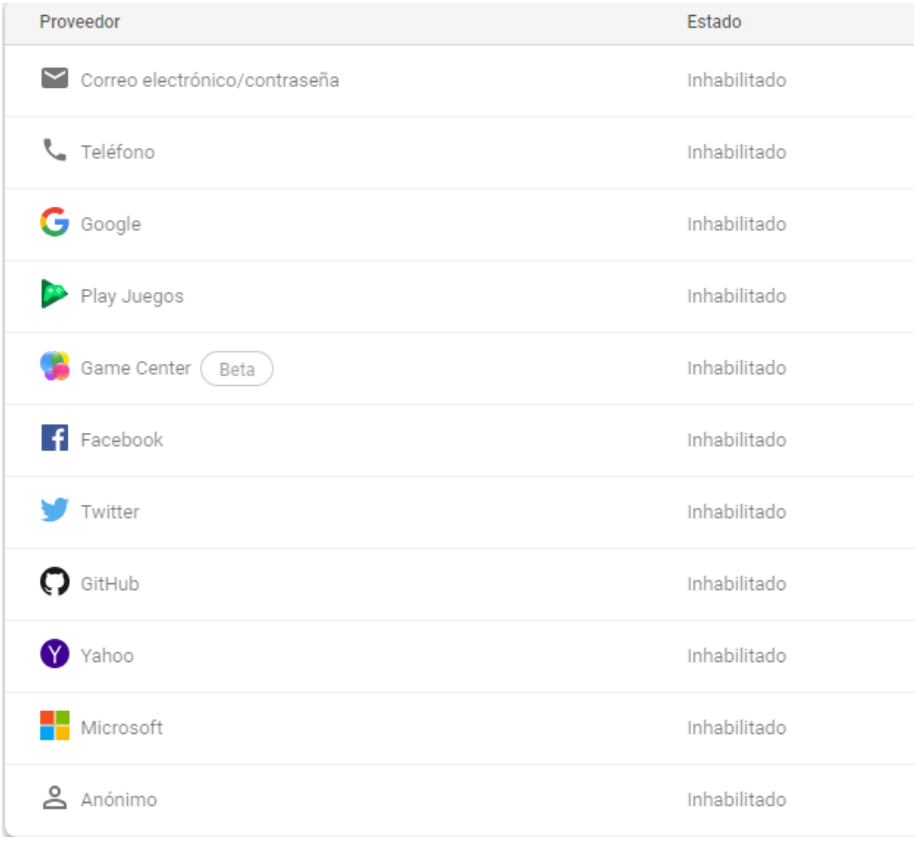


Ilustración 8: Posibles opciones de autenticación en Firebase

Como he comentado más arriba, la forma de autenticarse que elegí es a través de Google Play Juegos, que es amiga de Firebase, por ser también de Google y ofrece gran compatibilidad entre ambas.

3 Diseño

El proyecto al que deseaba llegar era uno que reuniera lo siguiente:

- Un sistema de autenticación para entrar a la aplicación, seguro, eficiente, único, online y fácil para el usuario.
- Una base de datos persistente, muy accesible y rápida.
- Un acuario bonito del que los usuarios se pudieran sentir orgullosos y percibir con claridad los cambios que en él se dan.
- Un sistema para detectar en qué medida se ha usado el móvil desde la última vez que se abrió la aplicación y un algoritmo para procesar esa estadística y convertirla en un cambio para el acuario.

Tras un periodo de investigación para descubrir las tecnologías anteriores y la organización de éstas, estructuré el proyecto en cuatro módulos diferenciados.

Esos módulos que componen el proyecto son los siguientes:

Módulo de Unity, que es el que dispone de la actividad principal de Android y el que realizará las llamadas al resto de módulos del proyecto, organizando al resto de módulos. Se ocupa de mostrar el acuario y gestionar los peces que lo componen, por ejemplo, moviéndolos. También mostrará un indicador con el progreso realizado desde la última vez que se abrió la aplicación para que el usuario pueda saber si lo está haciendo bien o no, y en qué medida.

Módulo de Android, encargado de que la aplicación disponga de los permisos suficientes para su cometido. Después se ocupa de las llamadas necesarias al sistema operativo para poder obtener el tiempo de uso del resto de aplicaciones. También procesa ese tiempo y lo normaliza a través de un algoritmo para convertirlo en un progreso razonable positivo o negativo. Este módulo, por tanto, dispone de la lógica base de la aplicación.

Módulo de Firebase, que se ocupa de almacenar los datos correspondientes a cada acuario de los usuarios y de mantenerlos actualizados cuando se aplique un progreso, de manera que el acceso al acuario se pueda realizar desde cualquier dispositivo y lugar con internet. Es el responsable de manejar esos datos para convertirlos en peces y viceversa. Todas las operaciones tienen que ser rápidas y eficientes, lo que se logra ideando bien la estructura de la base de datos.

Módulo de Google Play Juegos, que administrará inicios de sesión y registros a la aplicación, de manera que un usuario contará con la seguridad que ofrece Google para proteger su acuario. Además de contar con su eficiencia, su facilidad de uso y con el resto de las ventajas que ofrece la plataforma (como reunir todas las aplicaciones en una aplicación y guardar estadísticas sobre ellas, la posible incorporación de futuros logros, etc.).

4 Desarrollo

4.1 Módulo de Unity

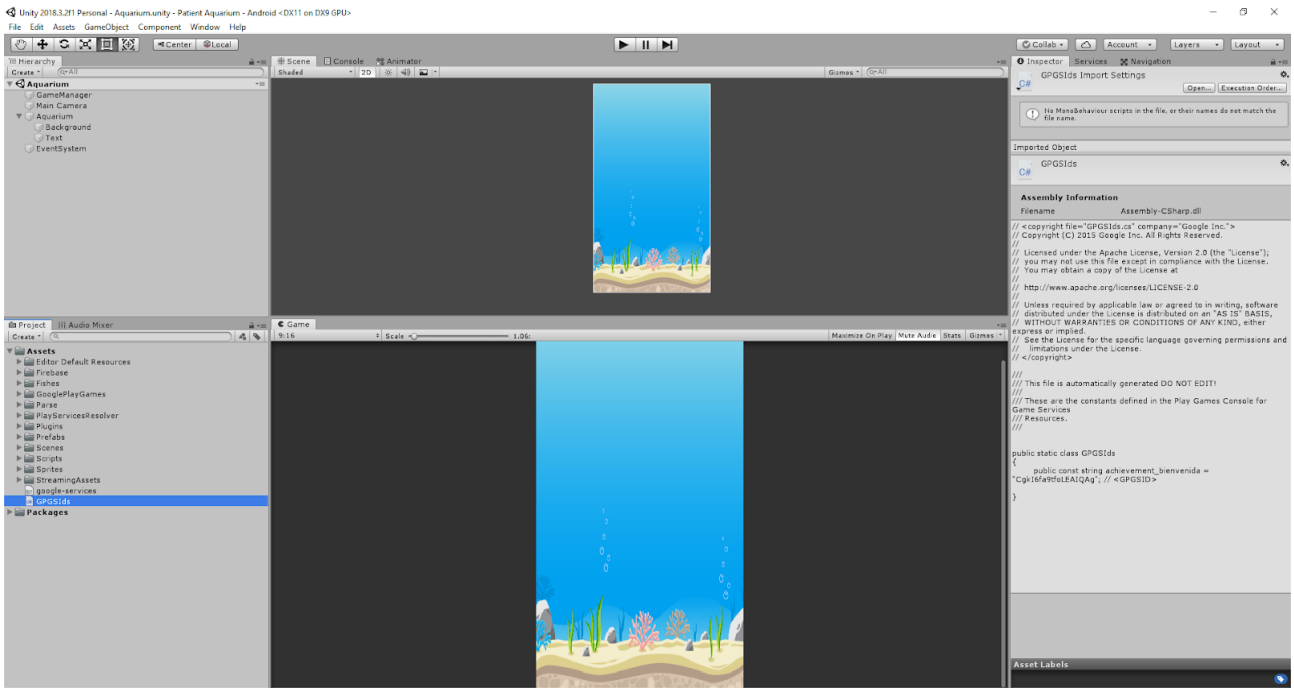


Ilustración 9: Entorno de Unity

En Unity se encuentra la actividad principal de Android de la aplicación y su correspondiente archivo de Manifiesto. También dispone de la pieza que conexiona los módulos entre sí. Su funcionalidad está dividida entre diferentes archivos y scripts:

- AndroidManifest.xml: En él se define la actividad que se lanzará al iniciar la aplicación. También gestiona los permisos de la aplicación: Que accederá a Internet y que se mostrará en la lista de aplicaciones capacitadas para que se les pueda otorgar manualmente permisos de acceso al tiempo de uso de otras aplicaciones. Otra de las tareas que realiza es establecer el icono y el tema de la aplicación.

- Gamemanager.cs: Es el script principal que se ocupa de hacer las llamadas al resto de scripts. Organiza el orden de ejecución de la aplicación y la estructura de ésta. Se inicia al arrancar la aplicación y se detiene al finalizarse. Dispone además de la función encargada de instanciar los peces. Para ello se comprueba el tipo de cada uno y en función de su nivel se le asigna un tamaño. Con esos parámetros establecidos se procede a la instanciación, que será en un lugar aleatorio dentro de la pantalla.

- Item.cs: Es la clase que contendrá los peces. Su nombre (objeto en español) se debe a que, en futuras versiones, podría haber objetos inertes por el mapa u otro tipo de criaturas acuáticas. La misión de este script es mover a los peces, cambiar su dirección, así como su ángulo cuando se llega a un extremo de la pantalla y sus

interacciones. También muestra y actualiza las barras de los progresos que han sufrido los peces cuando se inicia la aplicación.

4.2 Módulo de Android

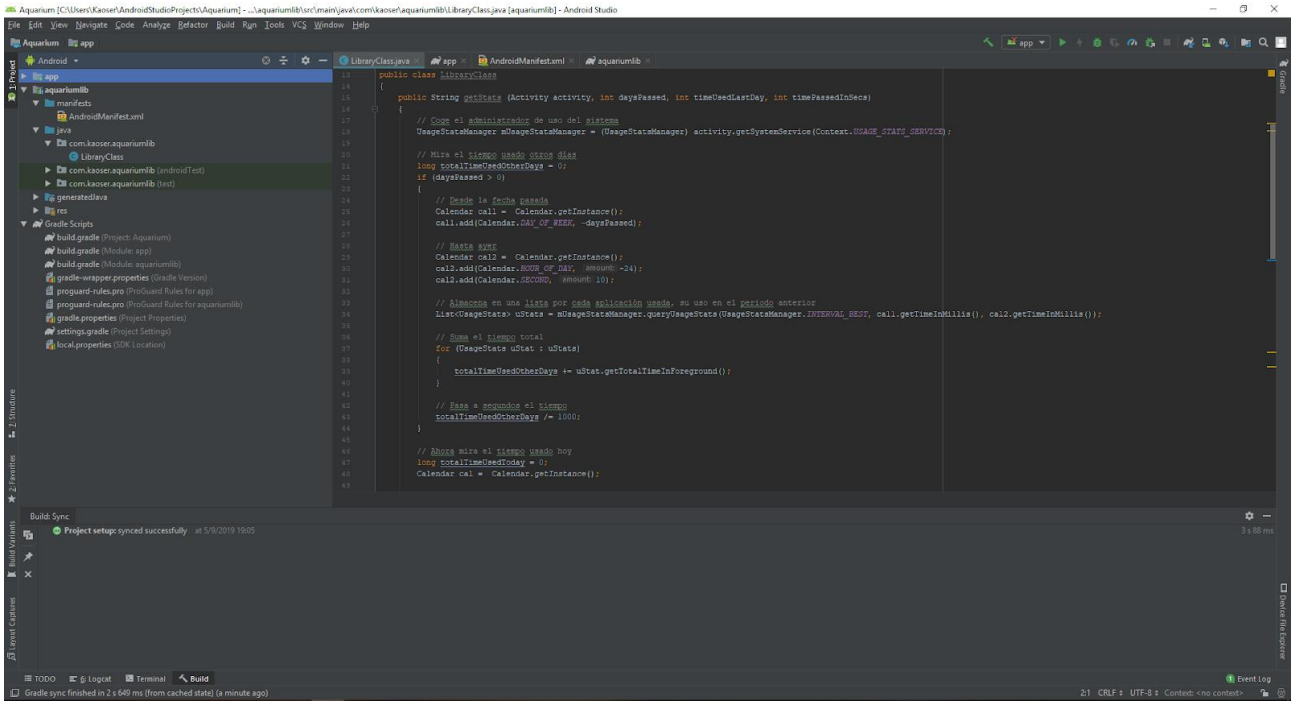


Ilustración 10: Entorno de Android Studio

Es el módulo desarrollado a través de Android Studio consistente en una biblioteca de Android llamada desde Unity. Su tarea será realizar las funciones específicas a las que Unity no puede acceder a través del lenguaje de scripting c#. Se compone de las siguientes clases:

- LibraryCaller.cs: Se ocupa de realizar desde Unity la llamada a la biblioteca de Android escrita en Java. Para ello obtiene la actividad de Unity (que es la única de la que dispone la aplicación) para pasársela a la biblioteca. Una vez tiene la actividad, llama a la función de la biblioteca pasándole como argumento esa actividad como "AndroidJavaObject", el tiempo transcurrido desde el último inicio de la aplicación en días y en segundos y el tiempo que se usó el último día antes de que se abriera la aplicación. Posteriormente se explicará la razón de cada argumento.

  Muchas de esas propiedades se calculan a partir de datos que LibraryCaller guarda en preferencias, para ser persistentes, ocultas e inmodificables por el usuario sin consumir el precioso espacio de la base de datos online. Los datos que se guardan en preferencias son el momento de la apertura de la aplicación, el tiempo que se ha utilizado hoy el móvil (información que se obtiene tras la llamada a la biblioteca) y el progreso decimal que no se puede aplicar hasta que sea entero, para tenerlo en cuenta en futuras iteraciones.

- LibraryClass.java: Esta biblioteca es la que contiene las llamadas necesarias al sistema operativo para acceder a la información de uso de otras aplicaciones.
  Lo primero que hace es comprobar que la aplicación disponga de permisos para realizar esta operación (que por defecto no está permitida). Si no los tiene, abrirá los ajustes en la zona de permisos de acceso al uso para que el usuario autorice a la aplicación.

  Ahora coge los argumentos que le pasa LibraryCaller en su invocación. Primero se utiliza la actividad para hacer la llamada al servicio del sistema encargado del servicio de estadísticas de uso. Este servicio genera estadísticas que acumula en bloques de intervalos de tiempo. El mínimo intervalo al que se puede acceder es de un día. Es por esto por lo que se pasa el número de días. Se recoge una lista de las aplicaciones usadas en el número de días que hayan pasado. El mínimo será un día. Después se recorre esa lista mirando el tiempo que esas aplicaciones han estado ocupando la pantalla en primer plano. Se suma el conjunto de tiempos y se convierte a segundos. Finalmente, se resta a ese tiempo el que usó el móvil el último día antes de que se abriera la aplicación. Con esto conseguiremos que no se contabilicen periodos ya contabilizados previamente.

  Al finalizar el proceso anterior se realiza el algoritmo pertinente a partir del tiempo pasado desde el último inicio de la aplicación y del tiempo de uso en ese periodo, y saca un progreso, positivo o negativo que se aplicará a los peces del acuario.

  El algoritmo: Hay varias posibilidades de progreso dependiendo del tiempo usado en el periodo pasado. Todas ellas son proporcionales a diferentes criterios:

  - Si el móvil se ha utilizado el tiempo proporcional a un minuto o menos por cada hora, el progreso será máximo y será equivalente a un punto de progreso por hora.
  - Si el móvil se ha utilizado el tiempo proporcional a dos minutos y medio o menos por cada hora, el progreso será equivalente a medio punto de progreso por hora.
  - Si el móvil se ha utilizado el tiempo proporcional a diez minutos o menos por cada hora, el progreso será equivalente a un décimo de punto de progreso por hora.
  - Si el móvil se ha utilizado el tiempo proporcional a media hora o menos por cada hora, el progreso será negativo y equivalente a medio punto de progreso por hora.
  - Si el móvil se ha utilizado más de media hora por cada hora, el progreso será máximo y de sentido negativo y equivalente a un punto de progreso por hora.

  De manera que favorece notablemente no usar el móvil o usarlo muy poco en un periodo. Por el contrario, penaliza un uso desmedido durante un periodo.

4.3 Módulo de Firebase

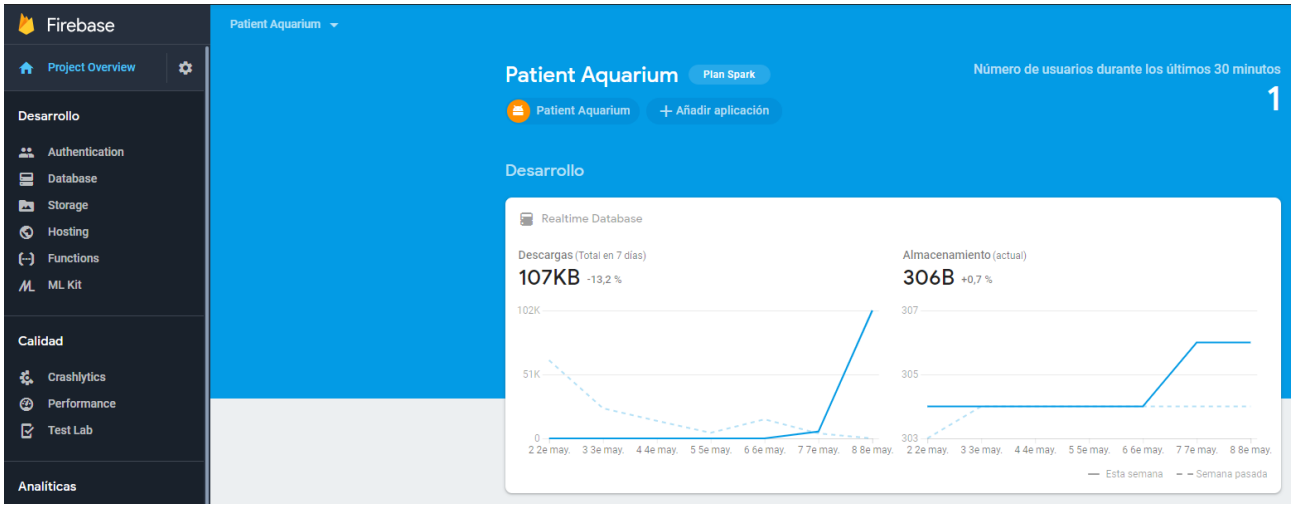
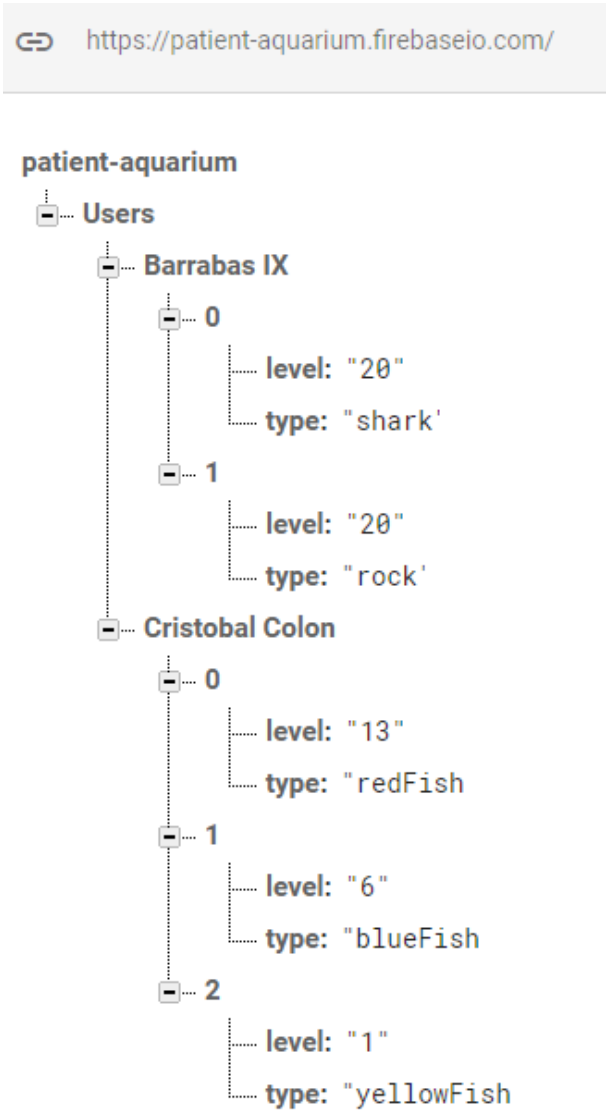


Ilustración 11: Página web de Firebase



La base de datos tiene una estructura muy sencilla y eficiente. Recordemos que Firebase trabaja con el formato JSON. Por tanto, la base de datos es un árbol que reúne a usuarios y sus acuarios y lo hace de la siguiente manera:

Sobre la raíz se despliega el nodo “Users” que es un diccionario con varias claves, los usuarios, y varios valores, sus acuarios. Cada usuario dispone a su vez de una lista de diccionarios, un elemento de esa lista será un componente del acuario. Esos componentes (que recordemos, son diccionarios) disponen de una clave para el tipo, con valor indicativo de la clase del objeto, como, por ejemplo, pez espada y una clave para el nivel con su valor de nivel correspondiente, que oscila entre 0 y 100 y determina el tamaño del componente.

- Aquarium.cs: Es el encargado de los accesos a los acuarios de los usuarios.
  Su funcionalidad es mirar la base de datos para obtener los peces de los que dispone el usuario, su tipo y su nivel para posteriormente mostrarlos. También actualiza la base de datos del usuario cuando hay cambios en el progreso de sus peces. La primera vez que se inicia la aplicación armará al usuario con un conjunto de cinco peces de nivel 25, para que pueda empezar a subir o bajar su nivel desde ese punto de partida.

4.4 Módulo de Google Play Juegos

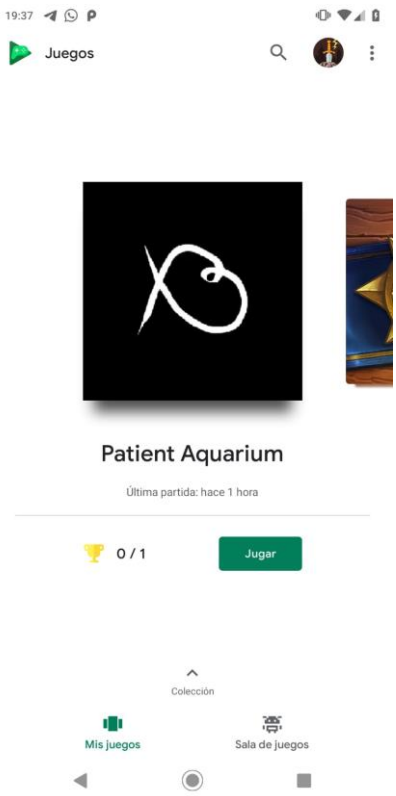


Ilustración 12: Aplicación Google Play Juegos

Es el módulo que realiza su cometido primero al iniciar la aplicación. Dispone de un solo archivo integrado en el proyecto de Unity que será el que realice los accesos al inicio de sesión y registro de los usuarios.

- SignIn.cs: Su funcionalidad reside en acceder a la API de Google Play Juegos, iniciar sesión en ella (o crear un usuario en caso de no tener cuenta) y solicitar un token para iniciar sesión posteriormente en Firebase Authentication. Si todo va bien, tras recibir el token se generan credenciales únicas para cada sesión para acceder a Firebase. Con esas credenciales se intenta iniciar sesión y, si todo sale correctamente, se continúa con la aplicación. Si el usuario ya dispone de cuenta de Google Play Juegos, el inicio de sesión no requiere de ninguna acción para el usuario y no necesita ni siquiera escribir sus datos para proceder en él.

  En caso de que algún paso del proceso falle, el juego se cerrará.

5 Integración, pruebas y resultados

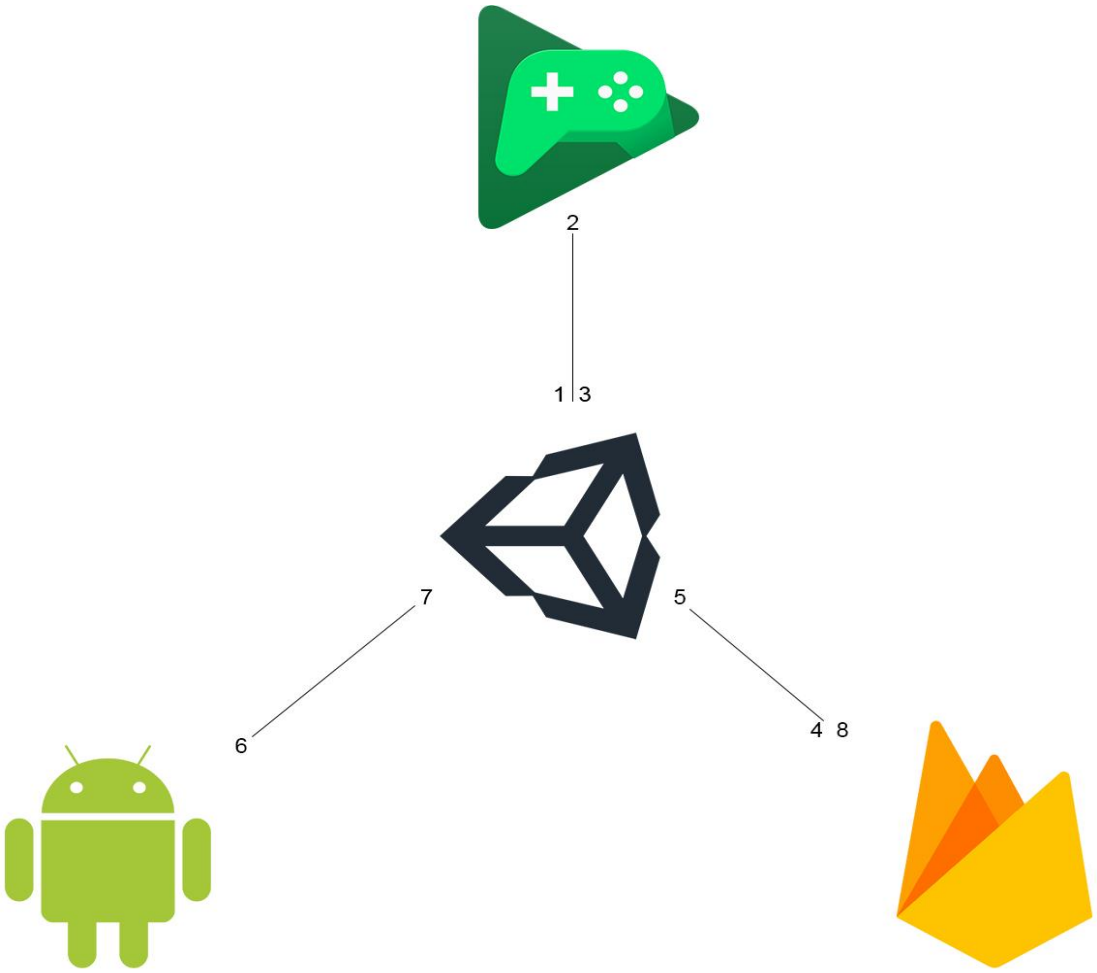


Ilustración 13: Interacción entre módulos

Para explicar cómo interaccionan los módulos entre sí es preciso explicar la ejecución del programa, que es como se detalla a continuación:

1. Lo primero que sucede cuando lanzas la aplicación es que Unity carga la actividad principal y prepara la ejecución de los scripts.

2. Al iniciarse la actividad se lanza el script de Unity encargado de iniciar sesión en Google Play Juegos. Si el proceso concluye de forma correcta, se procede al siguiente paso. En caso contrario la aplicación se cierra, ya que no podrá proceder a las sucesivas operaciones.

3. Ahora el GameManager que, recordemos, es el script que controla el flujo de ejecución de la aplicación y se encuentra en Unity, realiza una llamada al fichero Aquarium para que comunique con Firebase.

4. Firebase obtiene los peces de los que dispone el usuario. El resultado de la consulta se ordena en una lista de peces, que son diccionarios con tipo y

nivel, para que el siguiente proceso pueda continuar con el tratamiento de esos peces.

5. Después, el GameManager le cede el control a LibraryCaller, que ejecutará la biblioteca de Android.

6. La biblioteca accede al tiempo de uso de todas las aplicaciones del móvil desde la última vez que se abrió el acuario. Pero previamente comprobará que la aplicación dispone de permisos para hacerlo. En caso negativo se abrirá la ventana que facilita al usuario hacer que la aplicación obtenga esos permisos.

   Tras la obtención del tiempo total de uso del móvil en el periodo pertinente, se realiza el algoritmo y devuelve un progreso a LibraryCaller para se lo pase a GameManager.

7. GameManager recoge ese progreso y se lo aplica a cada pez disponible. Instancia los peces con su forma y tamaño y llama al script Item para que tome el manejo de cada pez individualmente.

   El script se ocupa de que los peces empiecen su ciclo de vida, desplazándose por la pantalla e interaccionando con los elementos como corresponda.

8. Finalmente, se guardarán los nuevos niveles de los peces a través del Aquarium en Firebase, para que estén actualizados la próxima vez que se abra la aplicación.

Ahora hablemos del desarrollo y las pruebas realizadas. Han sido tediosas, para qué negarlo. Hasta comenzar a ver resultados sobre el tiempo de uso del móvil pasaron varios meses. El primer módulo en crearse fue ese, el encargado de la obtención del progreso. Por suerte, Android Studio cuenta con un extraordinario sistema de depuración a través de ADB. Android Debug Bridge (ADB) es una herramienta que permite la comunicación con un emulador o un dispositivo Android conectado. Permite realizar diferentes acciones, como la instalación y la depuración de aplicaciones. Además, proporciona acceso a una consola Unix para ejecutar comandos en el emulador o dispositivo conectado a través de un proceso daemon.

Ilustración 14: Consola de ADB

La primera fase del desarrollo consistió en lograr acceder a las estadísticas. Una vez con esas estadísticas, se intentó operarlas de manera que dieran como resultado números aceptables para el progreso del acuario. Para ello hicieron falta más pruebas dejando entre ellas periodos de tiempo pequeños, o más grandes, como días o semanas para contemplar cómo se comportaba el algoritmo en diferentes situaciones.

Paralelamente se codificó y diseñó el acuario, de manera individual, de forma que los peces se movieran de lado a lado de la pantalla adaptándose al tamaño del dispositivo, y que los parámetros del tamaño estuvieran en un rango razonable. Considero a ésta la parte del desarrollo menos ardua y más gratificante.

El tema del inicio de sesión fue muy difícil de probar ya que al principio no funcionaba y había demasiados factores por los que esto podía estar ocurriendo. Tras montones de tutoriales, comprobaciones y horas de observación y dedicación por fin funcionó.

La base de datos, al ser de acceso asíncrono, produjo fallos por ejecutarse varias funciones al mismo tiempo. La solución a esto fue esperar a que terminaran los hilos asociados. Además, hubo que variar el diseño de ésta porque al principio era muy ineficiente, y para acceder al acuario de un usuario se recorrían todos los demás usuarios con sus acuarios. Hicieron falta cientos de accesos a ella tanto para obtener datos como para subir nuevos o actualizar los actuales hasta que quedó correctamente configurada.

Finalmente, la integración volvió a hacerse tediosa. Hubo que ir poco a poco comprobando a través de ADB los puntos hasta los que llegaba la ejecución satisfactoriamente. También se introdujeron nuevos cambios como el traslado del archivo de manifiesto a Unity, el nombre de usuario de Google Play Juegos que determinaba el nodo usuario de Firebase, el cambio del nombre del paquete de la aplicación para hacer compatibles Firebase y Google Play Juegos, etc.

Por suerte nunca ha habido un largo periodo de estancamiento en ninguna de las tareas y ha sido más o menos amena, aunque larga, la creación de la aplicación.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Estoy orgulloso de poder reconocer este proyecto como creación mía. Si bien es cierto que todavía necesita algunos retoques para resultar atractivo al público y llegar al mayor número de usuarios posible, la mayor parte del trabajo está completada y, sirve para lo que considero un bien general.

El tiempo es hoy en día nuestro más preciado recurso, y hay que protegerlo frente a las tentaciones del día a día. Creo que ésta es una buena forma de combatir una de las más fácilmente accesibles: el móvil. Es una apuesta arriesgada, pero de no ser así la recompensa no sería tan atractiva.

Cambiando de tercio, gracias a este proyecto he aprendido a utilizar numerosas herramientas activas hoy en día en el mercado y que me servirán probablemente en el futuro. Android, Unity, APIs de Google, … todas son increíblemente polivalentes.

Un aprendizaje que he extraído de este TFG es el de la planificación individual en un proyecto relativamente grande llevado por un solo colaborador. Hasta ahora en la carrera las prácticas han sido por parejas en su mayoría y de menor envergadura, pero esto ha supuesto un nuevo reto que me complace haber superado satisfactoriamente.

6.2 Trabajo futuro

Este TFG es más o menos completo en funcionalidad, pero mejorable en varios aspectos. Entre ellos los gráficos. Digamos que necesitaría una mano de pintura.

Mis principales ideas para añadir son las siguientes:

- Una lista de amigos para poder comprobar sus acuarios rápidamente y deleitar tu superioridad o inferioridad ante ellos. La competencia surge a partir de la comparación y por tanto soy partidario de favorecerla. Esta idea incluiría la adición de un botón que permitiera regresar a tu acuario.

- Crear una serie de logros para que los usuarios más atrevidos pudieran desbloquear criaturas u objetos únicos a partir de verdaderos retos. Por poner algún ejemplo, un reto podría ser estar durante un día completo sin usar el dispositivo, o utilizar el móvil menos de una hora cada día durante una semana.

- Añadir decorados en el acuario, para poder ajustar el acuario a tu medida y darle una personalización. Entre otros, estos decorados podrían ser anclas, algas, barriles, calaveras, corales, piedras.

- Una posibilidad que me he planteado también ha sido la de convertir la aplicación a fondo de pantalla. De esta manera no tendrías que pulsar sobre ella para abrirla y podrías disfrutar de tus peces en todo momento. La agonía sería terrible al verlos disminuir en tiempo real.

- Por descontado queda que una de las principales mejoras consistiría en crear nuevos tipos de pez más atractivos, más bonitos y quizá con animaciones o 3D. Cualquier mejora visual es bienvenida en el proyecto.

Estas y muchas otras serían las mejoras que realizaría en un futuro.

Referencias

[1] Gaurav, “Mastering Android context”, de freeCodeCamp, 5 de junio, 2018, https://medium.freecodecamp.org/mastering-android-context-7055c8478a22

[2] “Crear una biblioteca de Android”, Android Developers, https://developer.android.com/studio/projects/android-library?hl=es-419

[3] “Manifiesto de la app”, Android Developers, https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419

[4] “Anexo: Historial de versiones de Android”, Wikipedia, https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android

[5] “Funciones de Contextos”, Android Developers, https://developer.android.com/reference/android/content/Context.html#getSystemService(java.lang.String)

[6] “Unity (game engine)”, Wikipedia, https://en.wikipedia.org/wiki/Unity_(game_engine)

[7] Sgoliver, “Preferencias en Android I: Shared Preferences”, sgoliver.net, 144 de marzo, 2011, http://www.sgoliver.net/blog/preferencias-en-android-i-shared-preferences/

[8] Cosmos, “Google Play Games. A fondo”, Xatakandroid, 18 de mayo, 2013, https://www.xatakandroid.com/juegos-android/google-play-games-a-fondo

[9] Miguh Ruiz, “¿Qué es Firebase de Google”, OpenWebinars, 9 de agosto, 2017, https://openwebinars.net/blog/que-es-firebase-de-google/

[10] “Android Debug Bridge”, Android Developers, https://developer.android.com/studio/command-line/adb?hl=es-419

[11] Anchal Goel, “10 steps to publish your Android app on Google Play Store”, OodleTechnologies, 21 de octubre, 2015, https://www.oodlestechnologies.com/blogs/10-Steps-To-Publish-Your-Android-App-on--Google-Play-Store

[12] Alex Drozhin, “Permisos de aplicaciones en Android 8: Guía completa”, Kaspersky, 26 de septiembre, 2018, https://www.kaspersky.es/blog/android-8-permissions-guide/17037/

[13] Shane Richmond, Head of Technology, “Smartphones hardly used for calls”, The Telegraph, 29 de junio, 2012, https://www.kaspersky.es/blog/android-8-permissions-guide/17037/

Glosario

AAR Paquete de Android que almacena una biblioteca y puede ser usada por una aplicación Android.

ADB herramienta de comandos versátil que favorece la comunicación con una instancia de un emulador o dispositivo Android.

Android Studio Entorno de desarrollo oficial para la plataforma Android.

API Application Programming Interface en inglés, determina el acceso a funciones de un determinado software.

APK Paquete de Android que almacena una aplicación y permite su instalación.

Base de datos Conjunto de datos de un mismo contexto almacenados.

C# Lenguaje de programación orientado a objetos y desarrollado por Microsoft.

Certificación Declaración firmada digitalmente por una entidad que da valor a otras claves públicas de entidades ajenas.

Clave pública y privada Elementos usados en el cifrado asimétrico de forma que solo la clave privada en conjunto con la pública es capaz de descifrar.

Daemon Proceso informático que se ejecuta en segundo plano.

Depuración de programas Proceso de identificar y corregir errores de programación.

Huella digital Mecanismo para defender los derechos de autor.

Java Lenguaje de programación orientado a objetos pensado para ser ejecutado sobre una máquina virtual sin importar la arquitectura subyacente.

JSON Lenguaje de marcado y formato de texto simple para el intercambio de datos

Keytool Utilidad para manejar claves y certificados. Sirve para administrar el par de claves privada y pública, administrar certificados con la autenticación y uso de firmas digitales. Permite cachear claves públicas en forma de certificado.

Kotlin Lenguaje de programación orientado a objetos más moderno que Java y capaz de reemplazarlo que corre también sobre la máquina virtual de Java.

Manifiesto (Android) Archivo XML que define los componentes, propiedades y permisos de una aplicación.

OpenGL Especificación estándar escrita en c que define una API para crear gráficos 2D y 3D.

Photoshop Editor de gráficos rasterizados cuya finalidad es el retoque de fotografías y gráficos

PhysX Motor diseñado para llevar a cabo cálculos físicos.

Script archivo de órdenes o programa.

SHA1 Función usada en criptografía que produce a partir de una entrada un valor de 160 bits que generalmente se representa en hexadecimal.

SQL Lenguaje de programación específico para manejar bases de datos.

SQLite Sistema de gestión de bases de datos interna de Android que permite manejar datos en lenguaje SQL.

Token Clave que facilita el proceso de autenticación.

XML Lenguaje de marcado usado para almacenar datos.

Anexos

Manual de instalación

Manual de instalación: La instalación es realmente sencilla, pues la aplicación está disponible en Google Play como cualquier otra que conozcamos publicada. El nombre comercial que le he dado es “Patient Aquarium”. Y el logo, diseñado con Photoshop, el siguiente:

Ilustración 15: Logotipo de Patient Aquarium

El proceso de publicación no ha sido fácil, ha conllevado varios trámites tanto legales como “papeleos”, detalles de la aplicación, montones de parámetros de configuración, encuestas y algunos cambios como el nombre del paquete de la aplicación. Como ya comenté arriba, esto ha sido posible gracias a tener una cuenta de Google Play de desarrollador.

En resumidas cuentas, el proceso consiste en publicar un servicio de juego, que antes tenía en modo de pruebas para que funcionara la autenticación con Google Play Juegos. Para ello hay que añadir cierta información imprescindible como una página web a la que acceder para empezar a jugar, un icono o la descripción del juego.

Ilustración 16: Publicación en servicios de juego

Tras este proceso se puede proceder con la publicación de la aplicación a Google Play. Esta es una fase independiente de la anterior ya que una aplicación puede no estar publicada como servicio de juego.

Deberemos de nuevo especificar varias opciones y finalmente subir el archivo APK para que se pueda publicar.

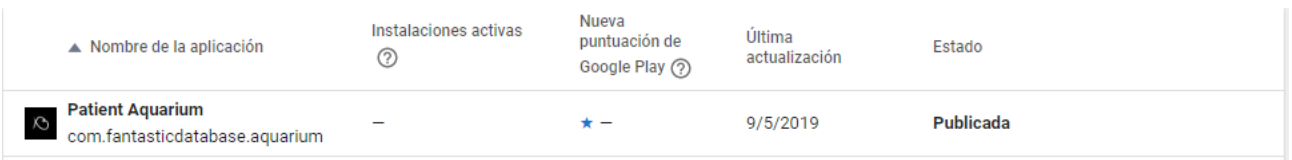


Ilustración 17: Publicación en Play Store

Manual de uso

En principio, el propósito de la aplicación consiste en no usar el móvil por lo que si no se utilizan ninguna de las funciones de éste (incluida, como es obvio, la aplicación), mejor que mejor.

Sin embargo, todos sabemos que esto es imposible así que voy a explicar cómo manejar la aplicación.

Es muy simple, solo necesitas abrirla y, en caso de ser la primera apertura, iniciar sesión en tu cuenta de Google Play Juegos o crear una en un proceso bastante simple y ameno, sin tener que desplazarte a ninguna URL.

Ilustración 18: Autenticación en Google Play Juegos

Tras finalizar el proceso, en caso de no tener permisos para el acceso al uso, la aplicación solicitará que se los des a través de la pantalla de ajustes correspondiente.

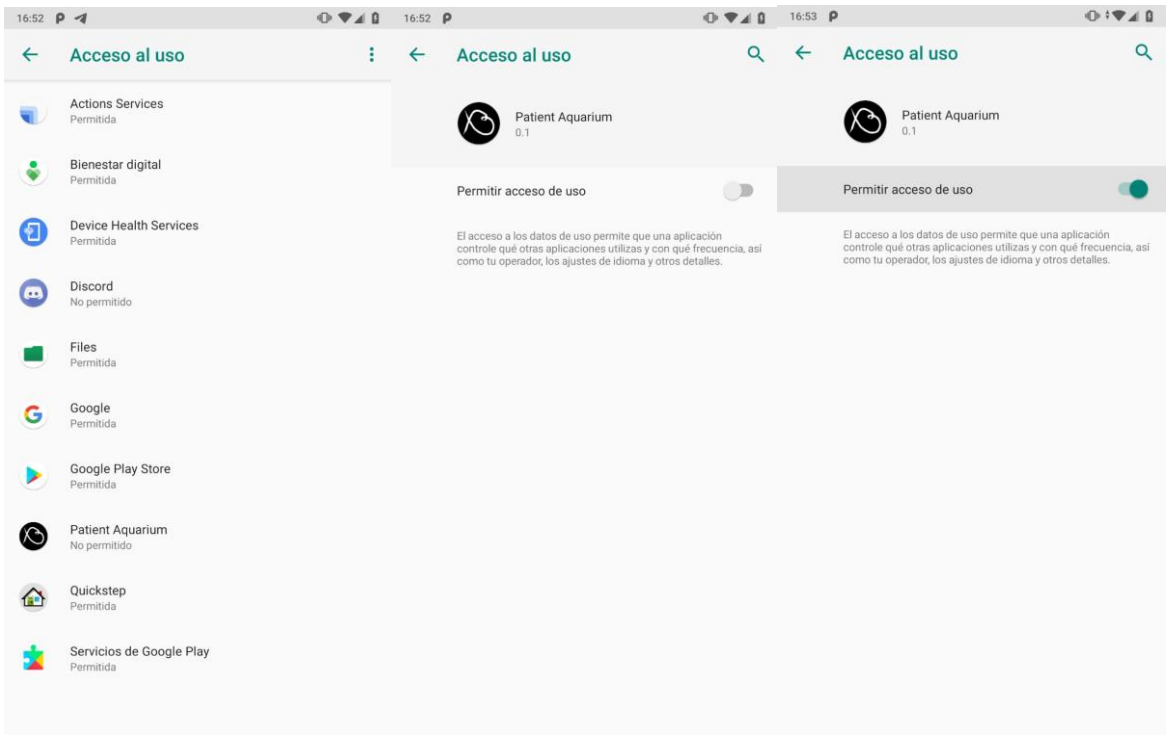


Ilustración 19: Asignación de permisos de uso

Finalmente, y en el resto de las ejecuciones, sólo deberemos y podremos observar. ¿Se trata de eso, no? No hay ninguna interacción disponible con el acuario para que no resulte entretenido ni merme nuestro valioso tiempo. Tan solo podremos observar cual ha sido el fruto de nuestras acciones entre la última apertura y esta.

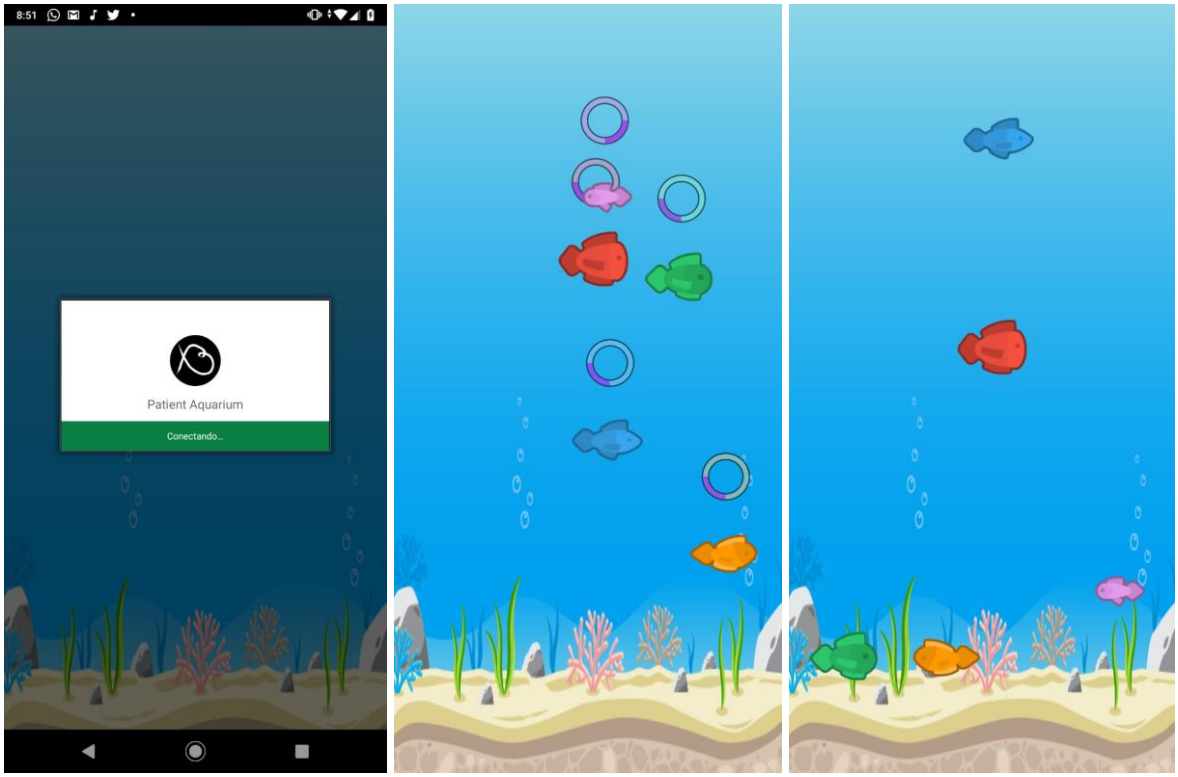


Ilustración 20: Patient Aquarium

Y después ver los peces moverse, actividad que no llevará al usuario más de un minuto, el usuario podrá cerrar la aplicación. Y a aprovechar el tiempo….